J.D. Lohn, J.A. Reggia, ``Discovery of Self-Replicating Structures using a Genetic Algorithm,'' 1995 IEEE International Conference on Evolutionary Computation, Perth, 1995, pp. 678--683.

Discovery of Self-Replicating Structures Using A Genetic Algorithm

Jason D. Lohn James A. Reggia
Departments of Computer Science and Electrical Engineering
A. V. Williams Bldg., University of Maryland, College Park, MD 20742
{jlohn, reggia}@cs.umd.edu

Abstract

Previous computational models of self-replication in cellular spaces have been manually designed, a very difficult and time-consuming process. This paper introduces the use of genetic algorithms to discover automata rules that govern emergent self-replicating processes. Given dynamically evolving automata, identification of effective fitness functions for self-replicating structures is a difficult task, and we give one solution to this problem. A model consisting of movable automata embedded in a cellular space is introduced and discussed in this context. A genetic algorithm using two fitness criteria was applied to automate rule discovery. After parameter tuning, 6 self-replicating structures consisting of 2, 3 and 4 automata were discovered over a course of 75 genetic algorithm runs. These results indicate that the fitness functions employed are effective and that genetic algorithms can be used to successfully discover rules for self-replicating structures.

1 Introduction

Studying computational models of self-replicating structures is a key area in the field of Artificial Life. Most of this work is based on cellular automata (CA), a model first used by von Neumann to study the complexity of self-replication. In a CA model, space is divided into cells, each containing a finite state machine. One state is typically designated the "quiescent" or inactive state, and the remaining states are said to be active. A self-replicating structure is represented as a configuration of contiguous active cells, each of which represents a component of the machine. At each discrete time-step, each automaton uses a set of rules to determine its next state as a function of its current state and the state of its immediate neighbor cells. Based solely on these concurrent local interactions, an initially specified structure goes through a sequence of steps to construct a duplicate copy of itself (the replica being displaced and perhaps rotated relative to the original).

Early studies focused on understanding the complexity of self-replicating structures in CA models. Researchers manually designed, presumably through trial and error, the CA rules needed to promote self-replication. Von Neumann was the first to report self-replicating structures in CA [17]. Using thousands of 29-state cells, his CA simulated a machine that could construct a copy of itself from instructions on a "tape." Subsequent research reported simpler self-replicating structures. Codd produced a sheathed loop structure embedded in an 8-state CA [4]. Langton made further reductions and described an 8-state, 86-component, sheathed-loop self-replicating structure [9]. Recently, even simpler, non-trivial self-replicating structures have been shown to exist [14]. In each of the above and related studies, automata rules were manually designed, and to our knowledge no research has been reported where such rules have been generated automatically. However, in other applications using rule-based automata models, some researchers have turned to genetic algorithms (GAs) as a method with which to search the space of CA rules [15, 8, 5].

The authors of the above studies reported encouraging results, and so we investigated whether GAs could be used for discovering rules for emergent self-replicating structures. The main barrier to this is the determination of effective GA fitness functions: given dynamically evolving automata, it is not obvious how to evaluate the development of a self-replicating structure. Using a rule-based automata model similar to CA models, we developed novel GA fitness functions and show for the first time that a GA could be used to discover rules that govern self-replicating structures.

2 Effector Automata

Many previous studies involving self-replicating automata structures were based on CA models. For our work, we developed a modified CA model

that retains desirable properties of CAs such as strictly local interactions among simple rule-based automata, emergent behavior, and the high degree of parallelism. Our model is called *effector automata* (EA). In both CA and EA models, a cellular space is defined where individual processing units (automata), operating in parallel, receive input from their local neighborhood, and using a predefined rule, produce an output. In CA models, each cell is an automaton; in EA models, each cell is a location in space, and automata are entities that can occupy cells. Thus, the two models differ in the kind of output produced: in CA models, the output is an internal state transition, whereas in EA, the output is an *action* to effect, such as moving to a neighboring cell. Automata models emphasizing actions, especially movement actions, have been investigated previously [1, 6, 13, 16, 3].

In the EA model used here, time is discretized, and space is an isotropic and infinite two-dimensional rectilinear grid composed of cells where a cell may be empty or occupied by an automaton a. The neighborhood template is the von Neumann neighborhood, and consists of five neighbors (including the automaton itself). Automata are directionally oriented: each distinguishes the relative locations, but not orientations, of its neighbors as top, right, bottom, and left. Each automaton is represented by a symbol in $\{\texttt{A}, \texttt{B}, \ldots\}$ indicating its *automata-type*, such that automata with the same automata-type use identical rules. A set A of N distinct automata-types is associated with an EA model, for example, with $N = 4$, $A = \{\texttt{A}, \texttt{B}, \texttt{C}, \texttt{D}\}$. The number of automatons of type a in a simulation at a given time-step t is called the *multiplicity* of a, and is denoted M_a^t.

The behavior of each automaton is governed by a *rule table*. Each rule table entry is a condition-action rule and has the form: `CTRBL` $\rightarrow$ *action* where `CTRBL` stands for center, top, right, bottom, and left neighbors. For example the rule, `B` • `A` • `C` $\rightarrow$ `DESTRUCT` specifies that if automaton `B` has `A` to its right, `C` to its left, and no others (• denotes an empty cell), then it should cease to exist at the next time-step. The actions used in the current EA model are listed in Table 1. Automata may move (both translation and rotation are included in the same action for convenience), divide into two copies (again movement is included for convenience), self-destruct, or remain inactive. Note that the `DIVIDE` action enables self-replication at the level of *individual* automatons, not the self-replication of multi-automata structures, the latter being the goal of this work. Values for the direction parameter (shown as `<dir>`) are either top, right, bottom, or left, and the rotation parameter (shown as `<rot>`) can be either 0, 90, -90, or 180 degrees. Because actions modify neighboring cells, a *collision policy* is specified to address the possibility of two or more automata attempting to occupy the same cell. Two example policies are *mutual annihilation* which results in all automata being destroyed, and the *random winner* policy which randomly selects one automata to occupy the cell in question. For this study, our EA model uses mutual annihilation.

3 Rule Discovery using a Genetic Algorithm

3.1 Genetic algorithm overview

We adapted a fairly standard genetic algorithm (GA) [7]. The important design parameters were: population size of 100 chromosomes, single point crossover and mutation operators, generational replacement with elitism, roulette wheel selection, and linear normalization of fitness scores. The GA was implemented in the C++ programming language and run on a Thinking Machines Inc. Connection Machine 5.

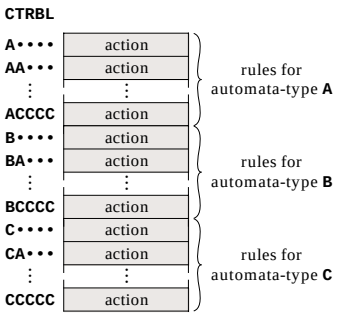


Figure 1: Chromosome representation in the GA.

The artificial chromosomes in the GA are comprised of EA rule tables. The representation chosen to encode an EA rule table for a three automaton-type system is depicted in Fig. 1. A simple table of condition-action rules is shown indexed implicitly by the neighborhood pattern `CTRBL`. All possible conditions are represented so that automata behavior is fully specified. A simulation consisting of N automata-types has $N(N+1)^4$ condition-action rules. Crossover and mutation operators manipulated the chromosomes as follows. After a random rule table index (crossover point) was chosen, an offspring chromosome was created by splicing together the resultant partial rule tables, one from each parent. Mutation operated by creating a random action for a randomly chosen rule table index. From experimentation, we found that a crossover rate of 0.8 and mutation rate near 0.1 yielded best

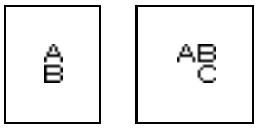
action	*description*
`MOVE <dir> <rot>`	move one cell in the specified direction and rotate the specified number of degrees
`DIVIDE <dir> <rot> <dir> <rot>`	divide into two daughter automata according to the specified directions and rotations
`DESTRUCT`	cease to exist
`NULL`	no action

Table 1: Actions used in current EA model

results.

3.2 Chromosome trials

The testing of each chromosome requires that a complete EA simulation be executed. For most EA simulations (and CA simulations), initial conditions play a critical role in determining system behavior. For our experiments concerning self-replicating structures, we started our system with small *seed* structures comprised of the two and three component automata structures shown in Fig. 2 a and b, respectively. The objective of the GA is to find a chromosome (rule table) that would allow the seed structure to produce copies of itself. We distinguish between trivial and non-trivial self-replication by insisting that the structure actively directs the construction of offspring, as opposed to trivial cases where all component automata simultaneously split to form two copies [9]. The duration or period T of each EA simulation was chosen to be $T = 10$ time-steps because given the small size of the seed structures, it was assumed that if any self-replicating processes emerged, it would be during these early time-steps.



(a) (b)

Figure 2: Seed structures.

3.3 Fitness functions

Designing a fitness function to evaluate self-replication is difficult because self-replication is a dynamic and complex process. Two key questions arise: what simulation data should be used and what criteria should be applied for an effective evaluation. Since the length of the desired self-replication cycle is unknown, using data from a single time-step would require knowing which time-step replicants appeared in and assumes that replicants appear all at once rather than at different time-steps. Clearly, data from multiple time-steps are needed so as to identify replicants as they are produced. The choice of which time-steps to use is important for similar reasons. Because our seed structures were very small, we chose to use the first 10 time-steps under the assumption that small structures would have fast replication cycles [14].

Since the locations and orientations of offspring from a self-replicating structure are not known before or during a simulation, comparing an evolving structure to a predefined template of seed copies by way of exact pattern matches fails to give partial credit during the replication cycle itself, when the structure has changed shape as it directs its self-replication. A better approach is to compare the adjacencies in the seed structure to the actual adjacencies seen at each time-step, giving partial credit for partial matches. This has the effect of guiding each automaton into the relative positions of the seed, hence producing a replicant. This was the approach we used.

Evaluating the fitness of each trial EA rule table and seed structure was accomplished by extracting statistics from each of the first 10 time-steps of an EA simulation. Using this data, two fitness measures were computed and then combined to give the overall fitness score, F, of each simulation. F was designed to reward continuous self-replication of the seed structure. Because there are two aspects to self-replication, F was expressed as the weighted sum of two normalized functions, a *growth measure*, f_{gm}, and a *relative-position measure*, f_{rpm}, as follows: $F = w_1 f_{gm} + w_2 f_{rpm}$, where $w_1 + w_2 = 1, w_1 \geq 0, w_2 \geq 0$. Growth measures assess production of individual automatons during the self-replication process. The growth measure used in this study was based on production rates. It measured to what degree each automata-type maintains a monotonically-increasing supply of offspring over time. Using f_{gm} to denote this particular growth measure, it is expressed as

$$f_{gm} = \frac{1}{2TN} \sum_{a \in A} \sum_{t=1}^{T} s_a(t) \qquad (1)$$

where

$$s_a(t) = \begin{cases} 2 & \text{if } M_a^t > M_a^{t-1} \\ 1 & \text{if } M_a^t = M_a^{t-1} \\ 0 & \text{if } M_a^t < M_a^{t-1} \end{cases} \quad (2)$$

The function $s_a(t)$ can be thought of as a scoring function that assigns point values on the basis of growth. For example, if there were 12 B automata at $t = 5$ and 14 at $t = 6$, then $s_B(t = 6)$ would be assigned 2 points. The summations in the function f_{gm} total the scores earned for each automata-type over each of the $T = 10$ time-steps. This sum is divided by the highest score possible to give a fraction representing how well monotonic production of automata progressed.

Given the state of an EA simulation at a particular time-step, relative-position measures (RPMs) assess the extent to which adjacencies between individual automata are similar to those in the seed structure. If the seed structure exhibits self-replication, then at certain time-steps individual automata will be positioned (relatively) just as they were in the seed structure, and a high RPM will be earned. Measure f_{rpm} relies on the following functions. The *seed neighbor count* function $\text{snc}(a)$ represents the number of occupied neighbor cells automaton a has in the seed. For example, in Fig. 2(b), $\text{snc}(\texttt{B}) = 2$, and $\text{snc}(\texttt{C}) = 1$. The *automata match* function $\text{am}(t, a)$ represents the number of neighbors of a at time t that were the same type and in the same relative position as in the seed. The *adjacency score* function $\text{adj}(t, a)$ is defined as:

$$\text{adj}(t, a) = \begin{cases} 0 & \text{if } M_a^t \leq 1 \\ \frac{\text{am}(t,a)}{M_a^t \cdot \text{snc}(a)} & \text{if } M_a^t > 1 \end{cases}$$

This represents to what degree, at time t, all the automata of automata-type a have the same neighbors as in the seed. When $M_a^t \leq 1$, automata a is extinct or is presumably part of the seed. When $M_a^t > 1$, $\text{adj}(t, a)$ is the ratio of $\text{am}(t, a)$ to the maximum possible. We then define f_{rpm} to be the mean of $\text{adj}(t, a)$ over all automata-types and all time-steps:

$$f_{rpm} = \frac{1}{T(N + (k-1))} \sum_{a \in A} \sum_{t=1}^{T} w_a \text{adj}(t, a). \quad (3)$$

where usually $w_a = 1$ ($a \in A$) and $k = 1$. In seed structures where one automata component a_l has more neighbors than the others (e.g., B component in Fig. 2(b) at t=0), it was advantageous to allow that automata to have more influence in the fitness calculation, and in that case we used $w_{a_l} = k = 2$.

3.4 Self-Replicating Structures

Six self-replicating structures were discovered in the course of 75 runs of 2000 generations each, where each run differed in the randomly-generated initial chromosomes. The first set of 11 experiments that led to three discovered self-replicating structures were performed using a 2-component seed structure with $w_1 = w_2 = 0.5$. For a 3-component seed, it was determined empirically based on preliminary runs that more emphasis was needed for relative positioning vs. component growth, and so we used $F = 0.2 f_{gm} + 0.8 f_{rpm}$. One 3-component and two 4-component structures were discovered over the course of 64 experiments started with 3-component seed structures. The data in Table 2 show the fitness values and the total number of self-replication rules for each of the discovered structures at generation 2000 (structures are named SRSxy, where x denotes the number of component automata of the structure and y is a suffix letter to make names unique). F values were in the range 0.004–0.009 for the first generation of randomly initialized rule tables, and increased to the values shown in Table 2 at the end of the simulations. Small numbers of rules were needed in each case, a result consistent with previous cellular automaton studies of self-replication [14].

structure	*rules*	f_{gm}	f_{rpm1}	F
SRS2a	8	0.944	0.885	0.914
SRS2b	21	1.000	0.612	0.806
SRS2c	12	0.861	0.761	0.811
structure	*rules*	f_{gm}	f_{rpm2}	F
SRS3a	13	0.741	0.810	0.796
SRS4a	15	0.815	0.887	0.872
SRS4b	8	0.926	0.869	0.881

Table 2: Rule and fitness data for 2-, 3-, and 4-component self-replicating structures.

As an example, the self-replication process discovered in this fashion for SRS3a is pictured and described in Fig. 3. It exhibits a four step replication process with offspring rotated 90 degrees clockwise relative to the parent. The original structure, and two offspring, can be seen in frame t=8. The subset of condition-action rules SRS3a actually used during self-replication are shown in Fig. 4.

4 Conclusion

The self-replicating structures produced in this fashion compare favorably in terms of simplicity with those generated manually in the past [14]. However, more interesting is that these replicating structures differed in unexpected ways from those developed in previous automata models. For example, they all were moving during replication, and all generated debris (unused extra components). In

```
A.... -> DESTRUCT                    B...A -> DESTRUCT
A.B.. -> DIVIDE RIGHT 0 BOTTOM 270   BA..A -> MOVE BOTTOM 90
A...B -> MOVE RIGHT 270              BAA.A -> MOVE LEFT 0
A.B.B -> DIVIDE TOP 90 RIGHT 0       B..AA -> DIVIDE LEFT 270 TOP 180
                                     BC.AA -> DIVIDE TOP 270 LEFT 180
CB... -> DIVIDE LEFT 0 RIGHT 0       B.AAA -> MOVE BOTTOM 90
C.B.. -> MOVE BOTTOM 90              B..CA -> DIVIDE LEFT 180 RIGHT 0
```

Figure 4: Subset of condition-action rules governing structure SRS3a (taken from the complete rule table).

some simulations, the replicant was not the initial seed structure but a larger structure built from it (SRS4a and SRS4b in Table 2). Such unanticipated results suggest that genetic algorithms, or other machine discovery methods, can be powerful tools for exploring the space of possible self-replicating structures. Furthermore, if the basic physical processes can be identified and represented effectively, such an approach might even be modified and applied to discover new self-replicating molecular structures.

References

[1] M. A. Arbib, Simple Self-Reproducing Universal Automata, *Inf. and Control*, **9**, pp. 177–189, 1966.

[2] A. Burks, *Essays on Cellular Automata*, Univ. of Illinois Press, 1970.

[3] H. Chou, J. Reggia, R. Navarro-González, & J. Wu, An Extended Cellular Space Method for Simulating Autocatalytic Oligonucleotides, *Computers Chem.*, **18**, 1, pp. 33–43, 1994.

[4] E. F. Codd, *Cellular Automata*, Academic Press, 1968.

[5] J. Crutchfield & M. Mitchell, The Evolution of Emergent Computation. Santa Fe Inst. Technical Report 94-03-012, 1994.

[6] N. S. Goel & R. L. Thompson, Movable Finite Automata (MFA): A New Tool for Computer Modeling of Living Systems. In [11], pp. 317–340, 1989.

[7] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, Mass, 1989.

[8] D. Jefferson, R. Collins, C. Cooper, M. Dyer, M. Flowers, R. Korf, C. Taylor, & A. Wang, Evolution as a Theme in Artificial Life: The Genesys/Tracker System. In [12], pp. 549–578, 1991.

[9] C. Langton, Self-Reproduction in Cellular Automata, *Physica D*, **10**, pp. 135–144, 1984.

[10] C. Langton, Studying Artificial Life with Cellular Automata, *Physica D*, **22**, pp. 120–149, 1986.

[11] C. Langton (ed), *Artificial Life*, Santa Fe Inst. Studies in the Sciences of Complexity, Vol. VI, Addison-Wesley, 1988.

[12] C. Langton, C. Taylor, J. D. Farmer, S. Rasmussen (eds), *Artificial Life II*, Santa Fe Inst. Studies in the Sciences of Complexity, Vol. X, Addison-Wesley, 1991.

[13] M. Lugowski, Computational Metabolism: Towards Biological Geometries for Computing. In [11], pp. 341–368, 1989.

[14] J. Reggia, S. Armentrout, H. H. Chou, & Y. Peng, Simple Systems That Exhibit Self-Directed Replication, *Science*, **259**, pp. 1282–1288, Feb., 1993.

[15] F. C. Richards, T. P. Meyer, & N. H. Packard, Extracting Cellular Automaton Rules Directly from Experimental Data, *Physica D*, **45**, pp. 189–202, 1990.

[16] I. Stephenson & R. Taylor, Creatures, A Simulation Environment for Autonomous Behavior. Technical Report ASEG.92.16, University of York (York, England, Y01 5DD), 1992.

[17] J. von Neumann *Theory of Self-Reproducing Automata*, A. Burks (ed), University of Illinois Press, 1966.

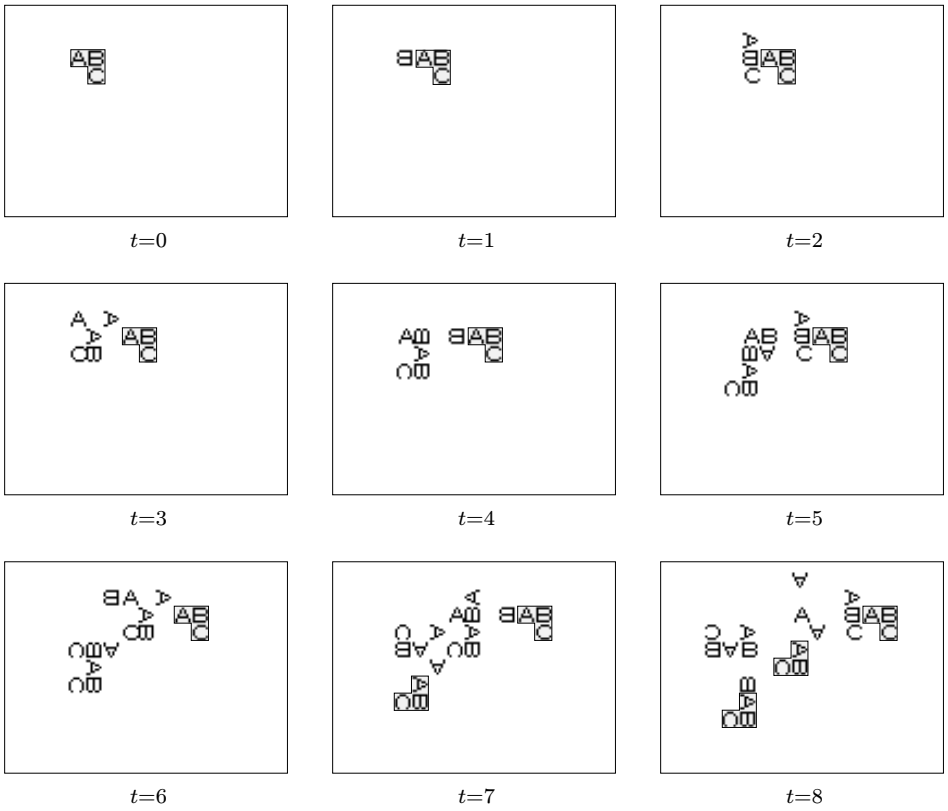


Figure 3: Development of 3-component self-replicating structure SRS3a. The original seed structure is highlighted, as well as the offspring, only after they appear in a clearly isolated position. The seed structure moves to the right at each time-step while producing offspring rotated 90° clockwise relative to the parent. The `B` component of the first offspring is produced (by division) at t=1, and the `A` and `C` components at t=2. The precursor of the first offspring is seen at t=3 and its `C` component rotates into position at t=4 as it continues to move downward. From t=5–7 it produces its own offspring and at t=7 it is seen isolated, and hence highlighted for the first time. The original seed structure produces its second offspring during t=4–7, which again moves downward. The original seed structure, its first two offspring, and an upside-down precursor offspring can be seen at t=8. Eventually, a diamond-shaped colony forms and expands indefinitely.